# Scaling Strong Consistency Across Continents with Hierarchical Consensus

*Omitted for review*


## Abstract

We introduce Hierarchical Consensus (HC), an understandable implementation and extension of Vertical Paxos designed to create large, strongly-consistent geo-replicated distributed systems. HC uses partitions across multiple independent subquorums to increase overall availability, localize decision-making, and improve throughput. Subquorums can be lightweight and chosen from hosts with low mutual communication latencies such that co-location and balanced load allow them to reach decisions quickly. Globally, a root quorum maintains the decision-space and provides linearizable guarantees across the entire system by intersecting with subquorums using a novel delegation approach to voting. We demonstrate HC's advantages through an implementation scaling to hundreds of replicas across more than a dozen availability zones around the world using Amazon EC2.


## Introduction

In this paper we tell the story of HC, a framework to create consensus across hundreds of replicas making decisions across an object space that is not independent and therefore requires strong consistency. We note that like in human societies, democracy works well in small groups but does not scale well to large groups, even for binary decisions (the e.g. the kind of decisions that distributed systems make). Inspired by modern governments, we propose a representative system of consensus, such that replicas elect leaders to participate in a root quorum that makes decisions about the global state of the system. Local decision making, the kind that effects only a subset of clients and objects is handled locally by subquorums as efficiently as possible.

As always, the devil is in the details. In this paper we first walk through the consistency model, then discuss the components of the system in an understandable way by using Raft as our base consensus algorithm and a key/value store as our primary application. Most of the rest of a paper is a discussion of safety. How can things go wrong in such a system? What are the implications? How can the system adapt, remain resilient, and even outperform other fault tolerant distributed consensus? We conclude with an evaluation and a discussion of related works ...

**problem**:

- standard consensus doesn't scale well
- it gets even worse across the wide area
- current systems assume object-independence
- current systems use fixed topographies and do not adapt

**motivation**:

- hierarchies already exist in applications
- geo-replication requires adaptivity
- transactions don't work well with tablet structures

**claims**:

- HC is an implementation and extension of Vertical Paxos [7] that organizes consensus hierarchically.
- Use Raft [11] for understandability
- No requirement for SQ protocol to be Raft, however there must be root quorum and subquorum intersections.
- we assert that leader-oriented protocols work best.
- generalized framework for scaling consensus

## Background

Paxos [5, 2, 6, 10, 1]

We implement and extend Vertical Paxos [7] but go beyond primary backup replication to creating a key/value store (and soon a file system).

Consider Niobe [9] and Boxwood [8] as two implementations of Vertical Paxos.

### Raft Consensus

A lightweight description of Raft and leader-oriented consensus protocols.

### Vertical Consistency Model

A command is identified by `(Epoch, Term, Log Index)` – mirrors Vertical Paxos `(Configuration, Ballot, Index)` structure.

Consistency: there must be a single, externalized ordering across all objects.

Figure: grid ordering consistency.

### Related Work

- ePaxos [10] is best geo-replicated consensus, doesn't scale.
- MDCC [4], Spanner [3] are big systems across multiple data centers but isolate objects into tablets that aren't coordinated.
- Niobe [9] and Boxwood [8] are Vertical Paxos for primary replication but treat configuration as an independent master quorum and have independent object management.

### HC Goals/Claims/Intuitions

**consistency claims:** We claim that for a current epoch, $E_i$, this externalized ordering exists for all epochs $\leq E_{i-j}$ such that $j \geq 1$ and $E_{i-j}$ is *closed*, e.g. a tombstone has been written to the logs of all subquorums, $S_{i-j}$. In the current epoch, $E_i$ it is not possible to externalize a complete ordering, however it is possible to describe an observable sequential ordering of all objects.

**performance claims:** we should make some.

## Design

Preliminaries: a system is composed of a set of replicas, $R$, initially we assume that this set is fixed, later we discuss adding or removing to this set. The system self organizes into a root quorum. The root quorum assigns replicas to zero or one subquorums.

During operation, a Replica can be *one* of the following.

1. A follower in a subquorum
2. A leader of a subquorum and a member of the root quorum
3. A hot spare

Replicas that are leaders can simultaneously be a subquorum leader and a root quorum leader. Later we relax this so that any replica can be a member of of the root quorum.

### Elections and Delegation

- election of root quorum leader
- assignment/election of subquorum leaders
- delegation of votes
- re-elections
- delegations expire

### Data Model

Ben: maybe move this to a different location?

- key/value store
- read then write updates
- accesses: get, put, delete

### Operation/Decision Making

- subquorum decision making
- remote reads and writes
- reconfiguration/epoch changes
- root quorum decision making
- transitions/fuzzy epochs

## Safety and Fault Tolerance

- individual faults
- partitions of the network (big faults)
- safety argument

## Implementation and Evaluation

- Go implementation
- AWS EC2 across 15 regions and 150 replicas (3, size 3 subquorums in 15 regions with 15 hot spares).
- LevelDB
- Version numbers

Evaluation

## Conclusion

- Psuedo code to make paper clearer
- Bobby: Provide top down description, where each model is relatively small, reduce cognitive load
- Bobby: Change name "nuclear option" – make more technical e.g. "final recovery" (too jocular)

## References


[1] M. Biely, Z. Milosevic, N. Santos, and A. Schiper. S-paxos: Offloading the leader for high throughput state machine replication. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium On*, pages 111–120. IEEE.

[2] L. J. Camargos, R. M. Schmidt, and F. Pedone. Multicoordinated paxos. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 316–317. ACM.

[3] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, and others. Spanner: Google's globally distributed database. 31(3):8.

[4] T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete. MDCC: Multi-Data Center Consistency. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 113–126. ACM.

[5] L. Lamport. Fast paxos. 19(2):79–103.

[6] L. Lamport. Generalized consensus and Paxos.

[7] L. Lamport, D. Malkhi, and L. Zhou. Vertical paxos and primary-backup replication. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, pages 312–313. ACM.

[8] J. MacCormick, N. Murphy, M. Najork, C. A. Thekkath, and L. Zhou. Boxwood: Abstractions as the Foundation for Storage Infrastructure. In *OSDI*, volume 4, pages 8–8.

[9] J. MacCormick, C. A. Thekkath, M. Jager, K. Roomp, L. Zhou, and R. Peterson. Niobe: A practical replication protocol. 3(4):1.

[10] I. Moraru, D. G. Andersen, and M. Kaminsky. There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 358–372. ACM.

[11] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319.